УДК 681.32

Ю.А. Скобцов, Д.Е. Иванов, А.И. Эль-Хатиб

РАСПРЕДЕЛЕННЫЕ ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ ГЕНЕРАЦИИ ПРОВЕРЯЮЩИХ ТЕСТОВ ЦИФРОВЫХ СХЕМ

Для повышения уровня “интеллекта” современных САПР компьютерных систем, расширения возможностей и повышения эффективности применяются различные методы искусственного интеллекта. Одним из самых перспективных направлений в генерации проверяющих тестов дискретных устройств является использование генетических алгоритмов (ГА) [1]. Данная работа продолжает исследования авторов по применению распределенных ГА (РГА) в генерации тестов, где на предыдущем этапе реализован РГА на основе модели «рабочий хозяин» [2].

генетические алгоритмы, генерация тестов, распределённые алгоритмы, моделирование с неисправностями

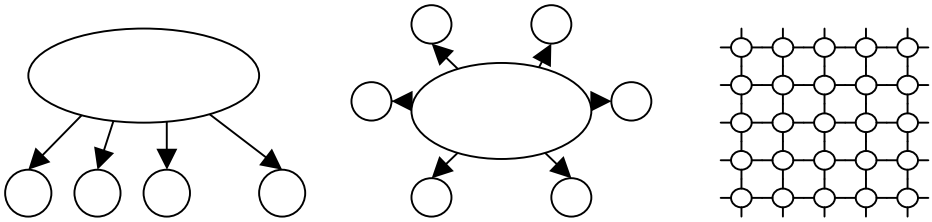
Генетические алгоритмы генерации тестов.

Целью автоматической генерации проверяющих тестов является построение входной последовательности двоичных наборов, которые проверяют любой физический дефект, возможный в процессе производства (или) эксплуатации логической схемы. Однако существует огромное число возможных потенциальных физических дефектов. Поэтому обычно рассматривают некоторый класс неисправностей, которые моделируют реальные физические дефекты. Таким образом, неисправность является моделью (одного или нескольких) физических дефектов. На практике чаще всего рассматривают одиночные константные неисправности, построение тестов для которых обычно дает удовлетворительные результаты. В случае необходимости (при низкой полноте тестов) тест достраивается для других типов неисправностей, таких как короткие замыкания, транзистор постоянно открыт (закрыт), задержки распространения сигналов и т.п. Известно, что генерация проверяющих тестов является NP-трудной задачей, т.е. в худшем случае решается путем перебора [1]. Поэтому, несмотря на то, что разработано множество последовательных структурных методов генерации тестов, которые для многих классов схем дают хорошие результаты, были предприняты попытки параллелизации алгоритмов построения тестов [3,4].

При генерации тестов для цифровых схем с применением ГА в качестве особи используется тестовая последовательность. Популяция состоит из конечного числа тестовых последовательностей, возможно, различной длины. Для выбранного таким образом представления особей и популяций применяются проблемно ориентированные генетические операторы кроссинговера и мутации [1].

Распределённые генетические алгоритмы.

Присущий ГА "внутренний" параллелизм и заложенная в них возможность распределенных вычислений способствовали развитию параллельных ГА (ПГА). Первые работы в этом направлении появились в 60-х годах, но только в 80-е годы, когда были разработаны доступные средства параллельной реализации, исследования ПГА приняли систематический массовый характер и практическую направленность. В этом направлении разработано множество моделей и реализаций, некоторые из которых представлены ниже[2].



а) б) в)

Рис.1. Различная реализация параллельных ГА.

Прежде всего, необходимо отметить, что в основе параллельных ГА лежит структуризация популяции (множества потенциальных решений) – его разбиения на несколько подмножеств (подпопуляций). Это разбиение можно сделать различными способами, которые и определяют различные виды ПГА. Согласно современной классификации различают Глобальные ГА (ГГА) распределенные ГА (РГА), клеточные ГА (КГА) и коэволюционные ГА (КЭГА).

Рассмотрим современные основные методы реализации ПГА. Наиболее известной является представленная на рис.1.а) ***глобальная параллелизация***. Эта модель основана на простом (классическом) ГА с вычислениями, выполняемыми параллельно. Она быстрее, чем классический ГА, выполняемый последовательно, и обычно не требует баланса по загрузке процессоров в системе, поскольку на разных процессорах чаще всего вычисляются значения фитнесс-функций для различных особей (имеющие примерно одинаковую вычислительную сложность). Исключение составляет генетическое программирование, где различные особи могут сильно отличаться по своей сложности (древовидные или граф- структуры). Данную модель часто называют **"рабочий - хозяин".** Многие исследователи используют пул процессоров для повышения скорости выполнения алгоритма, поскольку независимые запуски алгоритма на различных процессорах выполняются существенно быстрее, чем на одном процессоре. Отметим, что в этом случае нет никакого взаимодействия между различными прогонами алгоритма. Это чрезвычайно простой метод выполнения одновременной работы (если это возможно) и он может быть очень полезным. Например, данный подход может быть использован для решения одной и той же задачи с различными начальными условиями. В силу своей вероятностной природы ГА позволяют эффективно использовать этот метод. При этом ничего нового в сам алгоритм по сути ничего не вносится, но выигрыш во времени может быть значительным.

На рис.1 б) представлена также чрезвычайно популярная **"модель островов" (coarse grain)**, где множество подалгоритмов совместно работают параллельно, обмениваясь в процессе поиска некоторыми особями. Эта модель допускает прямую реализацию на компьютерных системах с MIMD-архитектурой. При этом каждый "остров" соответствует своему процессору.

РГА используют, в основном, так называемую "модель островов", где каждая подпопуляция развивается на своем "острове". Между островами производится (достаточно редко) обмен лучшими особями. Преимущество РГА в том, что они работают быстрее даже на однопроцессорных компьютерных системах вследствие лучшей структуризации. Причина заключается в том, что число вычислений сокращается благодаря распределению поиска в различных областях пространства решений. Разработаны различные виды РГА, но практически все они являются

вариациями базового алгоритма, который представлен следующим псевдокодом.

```
Распределенный ГА
{
 Генерация популяции P хромосом случайным
 образом();
 Разбиение P на подпуляции P1, P2, …,PN();
 Определение структуры соседства для SPi,
 I=1,…,N;
 While(критерий останова не выполнен)
  выполнение для SPi , I=1,…,N параллельно
  следующих шагов()
  {
   В течение fm поколений выполнение отбора и
   генетических операторов();
   Посылка nm хромосом в соседние подпопуляции;
   Прием хромосом от соседних подпопуляций;
  }
}
```

Основными факторами, которые влияют на миграцию в модели островов (и следовательно, на их эффективность) являются следующие.

1) ***Топология***, определяющая отношение соседства между подпопуляциями. Здесь обмен особями происходит только между соседними подпуляциями. Существует также несколько стандартных схем обмена особями между подпопуляциями, которые представлены ниже.

а) Каждая подпопуляция обменивается с каждой из остальных подпопуляций.

б) Обмен по кольцу.

в) Обмен между соседними подпопуляциями на гиперкубе.

2) Степень миграции, которая определяет количество мигрирующих особей.

3) Время изоляции, определяющее число поколений между сеансами миграции.

4) Стратегия отбора особей в пул обмена. Здесь наиболее распространенными являются два метода. При первом подходе из подпопуляции особи выбираются случайным образом – при этом сохраняется разнообразие генетического материала. При втором методе из каждой подпопуляции выбираются лучшие в некотором смысле особи, что делает процесс более направленным. При этом также существуют различные методы отбора лучших хромосом.

5) Стратегия замены особей на мигрировавшие хромосомы из соседних подпопуляций. Здесь также существуют различные подходы: из подпопуляции удаляются худшие, случайные особи и т.п.

6) Стратегия репликации мигрирующих особей. При первом подходе мигрирующая особь остается также и в "родной" подпопуляции. Второй подход требует удаления мигрирующей особи из "родной" подпопуляции. Впервая стратегия может привести к доминированию в различных подпопуляциях одних и тех же сильных особей. При второй стратегии особь может через некоторое время вернуться назад в исходную подпопуляцию, что ведет к лишним затратам вычислительных ресурсов.

Программная реализация.

Разработаны распределенные генетические алгоритмы генерации тестов, которые реализованы на основе локальной сети (обычного учебного компьютерного класса) программно с технологией блокирующих сокетов в среде программирования C++ Builder 6. Технология блокирующих сокетов означает, что процесс вычислений в точках приёма информации будет остановлен до тех пор, пока передающая сторона не отправит необходимые данные. Полученные экспериментальные данные для схем международного каталога ISCAS89 показывают, что данный подход на основе «модели островов» позволяет существенно повысить качество (полноту покрытия неисправностей) проверяющих тестов в отличие от модели «рабочий-хозяин», где в основном происходит

Таблица 1

Характеристика	Однопроцессорная реализация	Многопроцессорная реализация (число процессоров)					
		1	2	3	4	6	8
длина теста	1000	1000					
общее время моделирования, сек.	330	336 (0,98)	194 (1,7)	138 (2,39)	107 (3,08)	86 (3,83)	79 (4,17)
общее число событий, млн.	441,51	440,05 (1,00)	441,81 (1,00)	443,21 (1,00)	443,78 (1.01)	447,79 (1,01)	449,93 (1,02)
число событий моделирования исправной схемы, млн.	0,48	0,48 (1,00)	0,95 (1,98)	1,42 (2,96)	1,90 (3,96)	2,85 (5,93)	3,80 (7,92)
число событий моделирования схем с неисправностями, млн.	441,03	439,58 (1,00)	440,86 (1,00)	441,79 (1,00)	441,88 (1,00)	444,94 (1,01)	446,13 (1,01)

ускорение процесса построения тестов при сохранении (или небольшом улучшении) качества.

Разработанное программное обеспечение апробировалось на схемах из международных каталогов ISCAS-85 и ISCAS89, на которых по международным стандартам принято испытывать новые методы генерации тестов.

Для проведения экспериментов использовался вычислительный кластер, построенный на базе локальной сети 100 Мбит одной из учебных аудиторий. Элементы кластера, включая сервер, имеют следующие характеристики: процессор Intel Celeron 2000Мгц, объём ОЗУ 256Мбайт, операционная система Windows XP.

С целью изучения условий эффективного применения предложенного алгоритма, в процессе проведения машинных экспериментов проводился замер следующих временных характеристик: общее время процесса моделирования, число событий при моделировании исправной схемы и схем с неисправностями, общее число событий. Для сравнительной базы бралась работа алгоритма из [5] на персональном компьютере соответствующей конфигурации.

В табл 1 приведены числовые данные при проведении машинных экспериментов со схемой средней размерности S9234.ben. Данная схема имеет следующие характеристики: число входов – 19, число выходов – 22, число D-триггеров – 228, число инверторов – 3570, число вентилей других типов – 2027, общее число элементов – 5866. В таблице в скобках при многопроцессорной реализации указано увеличение быстродействия (для времени моделирования) и измеряемого параметра при сравнении с однопроцессорной реализацией алгоритма [5].

При анализе видно, что число событий при моделировании работы схем с неисправностями практически не растёт (коэффициент ≈1). Это говорит о том, что при предложенной схеме работы алгоритма распараллеливания отсутствует избыточное моделирование схем с неисправностями. С другой стороны, число событий при моделировании работы исправной схемы растёт с коэффициентом близким к числу задействованных процессоров: от 1,00 до 7,92 при увеличении числа процессоров-клиентов от 1 до 8.

Это обусловлено необходимостью моделирования работы исправной схемы на всех процессорах-клиентах. Общее же число событий растёт несущественно: коэффициент изменяется от 1,00 до 1,02 при увеличении числа процессоров-клиентов от 1 до 8. Это связано с тем фактом, что число событий при моделировании исправной

схемы составляет менее 1% от общего числа событий.

На рис.2 приведен график роста быстродействия при изменении числа процессоров-клиентов от 1 до 8 по данным моделирования схемы S9234.ben. Для сравнения на диаграмме присутствует график линейного роста быстродействия с коэффициентом равным 1.

Приведённые экспериментальные данные подтверждает эффективность предложенной схемы параллелизации алгоритма моделирования. Однако полностью линейного увеличения быстродействия достичь не удаётся. Основным препятствием для этого являются необходимость многократной (по числу задействованных процессоров-клиентов) передачи описания схемы, а также избыточное моделирование работы исправной схемы на каждом рабочем процессоре.

На рис 3 представлены данные моделирования с неисправностями также для большой схемы каталога ISCAS 89, которые также показывают зависимость скорости моделирование от числа процессоров-клиентов. Приведенная кривая подтверждает приведенные выводы. Наконец, на следующем рис.4 представлены данные моделирования для одних из самых больших схем каталога ISCAS89. Эти данные показывают, что

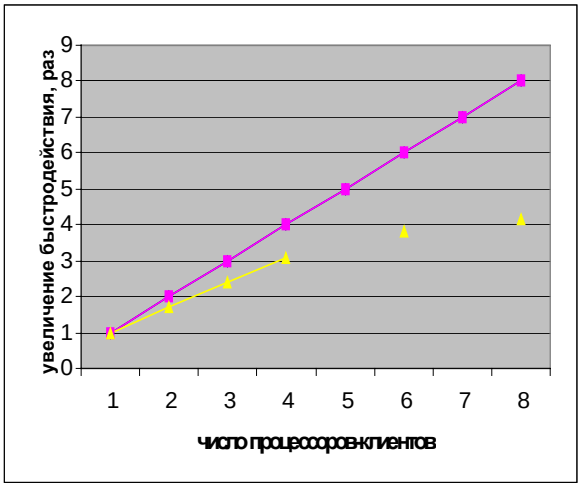


Рис.2 Рост быстродействия при увеличении числа процессоров-клиентов.

наблюдается увеличение относительной скорости моделирования неисправностей с ростом размера схемы. Это можно объяснить тем, что для больших схем “накладные расходы” по организации распределенного моделирования становятся меньше по сравнению с затратами непосредственно на моделирование неисправностей..

Характеристики полученных тестовых наборов представлены в табл.2

Выводы

Программно реализован модули распределенного моделирования и генетического алгоритма построения тесто на основе модели

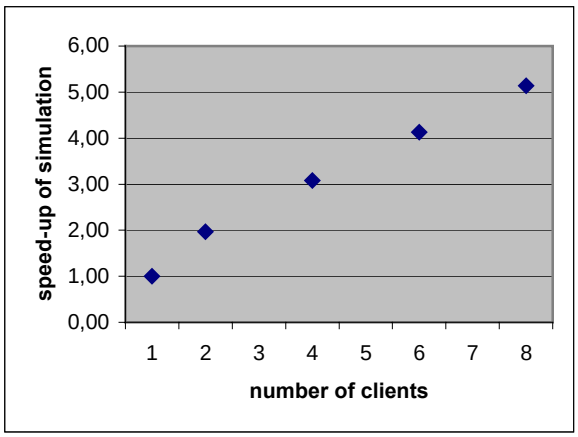


Рис.3 Увеличение скорости моделирования для схемы s35938 в зависимости от числа клиентов.

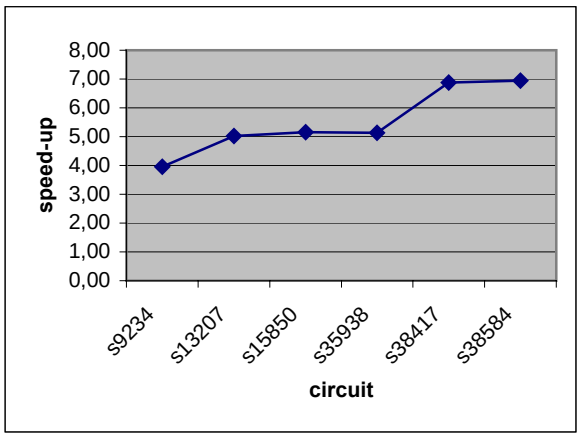


Рис 4 Увеличение скорости моделирования с ростом размера схем (для больших схем каталога ISCAS-89) для 8 клиентов.

Таблица 2

Имя схемы	Число Неисправностей	Длина теста	Полнота теста, %	Условная полнота теста, %
S298	308	637	82.79	85.39
S344	342	192	96.20	97.95
S349	350	295	95.71	97.43
S386	384	606	68.49	68.49
S641	467	727	86.30	87.79
S713	581	677	80.38	81.76
S1196	1242	1642	94.77	94.77
S1238	1355	1183	90.18	90.18
S1488	1486	1975	70.46	70.59
S1494	1506	1712	70.58	71.18
S3271	3270	1125	97.98	97.98
S5378	4603	120	71.73	73.66

“рабочий-хозяин”, который интегрирован с систему моделирования и генерации тестов АСМИД. Проведена апробация и проверка эффективности разработанных программных модулей на логических схема из международных каталогов ISCAS-85 и ISCAS89. Экспериментально доказана целесообразность проведения распределенных вычислений; время затраченное на синхронизацию и обмен данными между подпопуляциями, для предложенных параметров метода, не превысило 15% общего времени поиск решения. При этом общее время решения сократилось на величину от 800% до 50%, в зависимости от размерности обрабатываемых схем. Исследованы временные характеристики работы алгоритма. Экспериментально показано, что предложенный алгоритм построения проверяющих тестовых последовательностей улучшает характеристики автоматизированной системы моделирования и диагностики.

Полученные экспериментальные данные для схем международного каталога ISCAS89 показывают, что подход на основе «модели островов» позволяет повысить качество (полноту покрытия неисправностей) проверяющих тестов в отличие от модели «рабочий-хозяин», где в основном происходит ускорение процесса построения тестов при сохранении (или небольшом улучшении) качества.

Литература:

1. Ю.А.Скобцов, В.Ю.Скобцов. Логическое моделирование и тестирование цифровых устройств.-Донецк:ИПММ НАНУ, ДонНТУ, 2005.-436с.
2. Д.Е.Иванов, Ю.А.Скобцов, А.И.Эль-Хатиб. Распределенные алгоритмы моделирования и генерации тестов // Радіоелектронні і комп’ютерні системи.-ХАІ:2006.-№6.-с.97-102.
3. D.Krishnaswamy, M.Hsiao, V.Saxena, E.M.Rudnick, J.P.Patel. Parallel genetic algorithms for simulation-based sequential circuit test generation // IEEE VLSI Design Conference, 1997.- pp.475-481.
4. F.Corno, P.Prinetto, M.Rebaudengo, M.Sonza Reorda, E.Veiluva. Aportable ATPG tool for parallel and distributed systems // Proc VLSI Test Symp 1996.-pp.29-34.
5. Иванов Д.Е., Скобцов Ю.А. Параллельное моделирование неисправностей для последовательностных схем // Искусственный интеллект. – 1999. - №1. – С.44-50.